

Investment Explorer

A first touch automated response to Novice
Investor Queries

Technical Report

Prepared by Sheila O'Donnell
May 12, 2012

1 Abstract

Meetings between financial advisors and their clients are often taken up with explanations about straight-forward financial terms. While these terms may be straight-forward to the financial advisor, they are not for the client. This application gives the client a medium through which to gather information but frees the financial advisor as the answer need only be given once to populate the knowledge base. After that, the application responds to the client.

The application uses an AIML engine to try to mimic a human response to the questions asked. Communication is via email.

The interaction is simple but a useful starting point for the user.

1	Abstract.....	2
	Introduction.....	4
1.1	Background	4
1.2	Aims.....	4
1.3	Technologies.....	4
1.3.1	Java in Eclipse IDE.....	4
1.3.2	JUnit	5
1.3.3	Spring	8
1.3.4	Ant	8
1.3.5	ProgramD	8
1.3.6	AIML	9
1.3.7	MySQL	9
1.3.8	IMAP/SMTP	9
1.3.9	Tomcat.....	10
2	System.....	11
2.1	System Architecture	11
2.2	Process Flow Diagram	12
2.3	Code base Structure	13
2.4	Requirements	13
2.4.1	Functional Requirements	13
2.4.2	Data Requirements	15
2.4.3	User Requirements	15
2.4.4	Environmental Requirements.....	16
2.4.5	Usability Requirements.....	16
2.5	Evaluation	17
2.5.1	Unit Testing	17
2.5.2	System Testing	17
3	Conclusions & Recommendations	20
3.1	Application Improvements	20
3.2	Implementation Improvements	20
4	Bibliography.....	21
5	Appendix	22
5.1	Database create script	22
5.2	Ant build file	22
5.3	Ieexplorer-start.bat	29
5.4	AIML generator bat.....	30
5.5	AIML files	30
5.6	Project Proposal	30
5.7	Project Plan	35
5.8	Requirements Specification.....	36

Introduction

1.1 Background

It can be difficult to find consistent information about financial products and services in Ireland. In particular, many terms that are used on foreign publications are not used here and therefore difficult to understand.

The application already has a foundation of useful terms built in but an experienced business user can improve the service by adding additional terms and publishing to the application.

Users send emails asking questions which are answered by the application using the knowledge base.

1.2 Aims

- Seamless communication between the user and the application
- Give human-like responses
- Easy to support the application and update the knowledge base

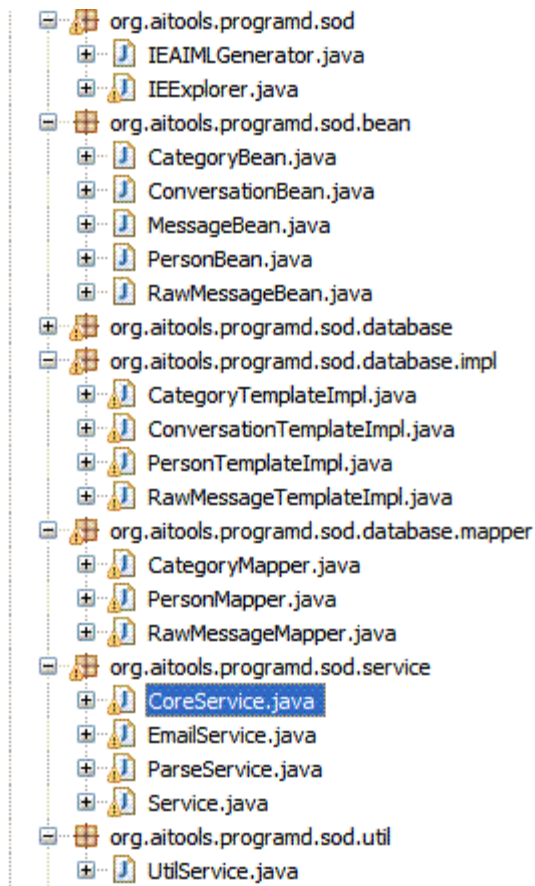
1.3 Technologies

1.3.1 Java in Eclipse IDE

Java in Eclipse was chosen as the main programming language because it encourages writing modular code. This in turn facilitated writing effective test scripts. It is a mature programming language with significant support forums available.

The screenshot below shows the package structure where the modular nature of the project can be clearly seen.

Breaking the code in to modules allows it to be easily understood and maintained.



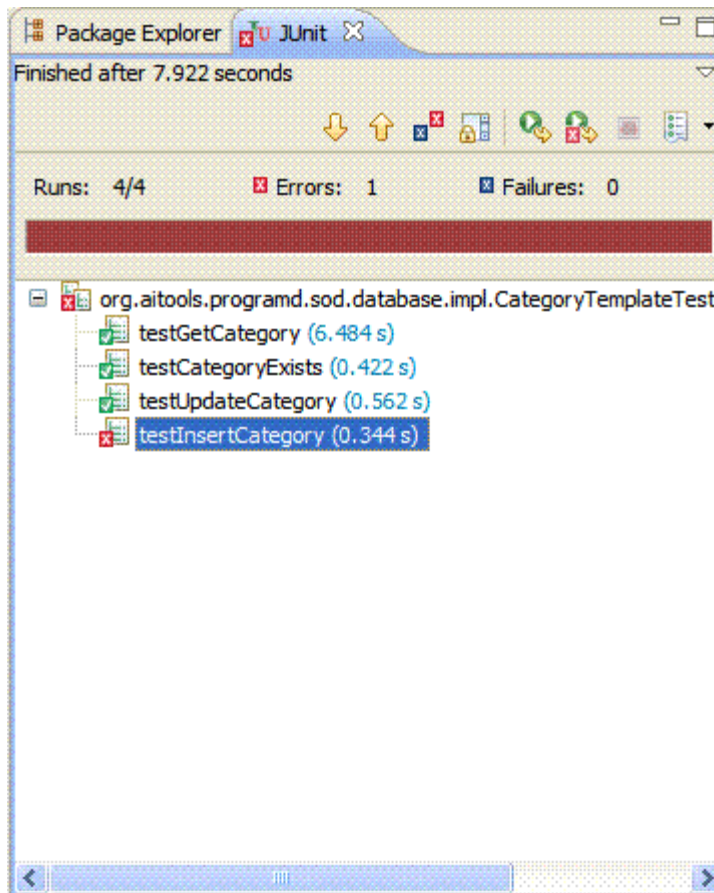
1.3.2 JUnit

1.3.2.1 Overview

As part of the evaluation process, I needed to write effective test scripts. JUnit provides a systematic approach to testing and integrates well with Eclipse. It provides a method by which the developer can verify that the latest changes haven't broken the build.

The figure below shows the impact of a change made to the code base that has caused a test to fail. By running the test, the developer can determine what impact his change has. It is an early warning that the build may be broken.

This is particularly useful when working in a group on a large code base as it deters a developer from checking in a change that fails the JUnit test.



1.3.2.2 JUnit Example

The following is an example of the type of tests that were created in order to test and regression test the software.

While parsing the users email, we need to identify and present questions to the knowledge base. The following test verifies the `getQuestion()` method of the `ParseService` class.

We present 'question' and 'noQuestion' to the method. If the class returns the expected response, the test will pass. If not, it will fail.

```

/**
 * Identify questions
 */
@Test
public void testGetQuestion() {
    String question = "Am I a question?";
    String noQuestion = "I am not a question.";

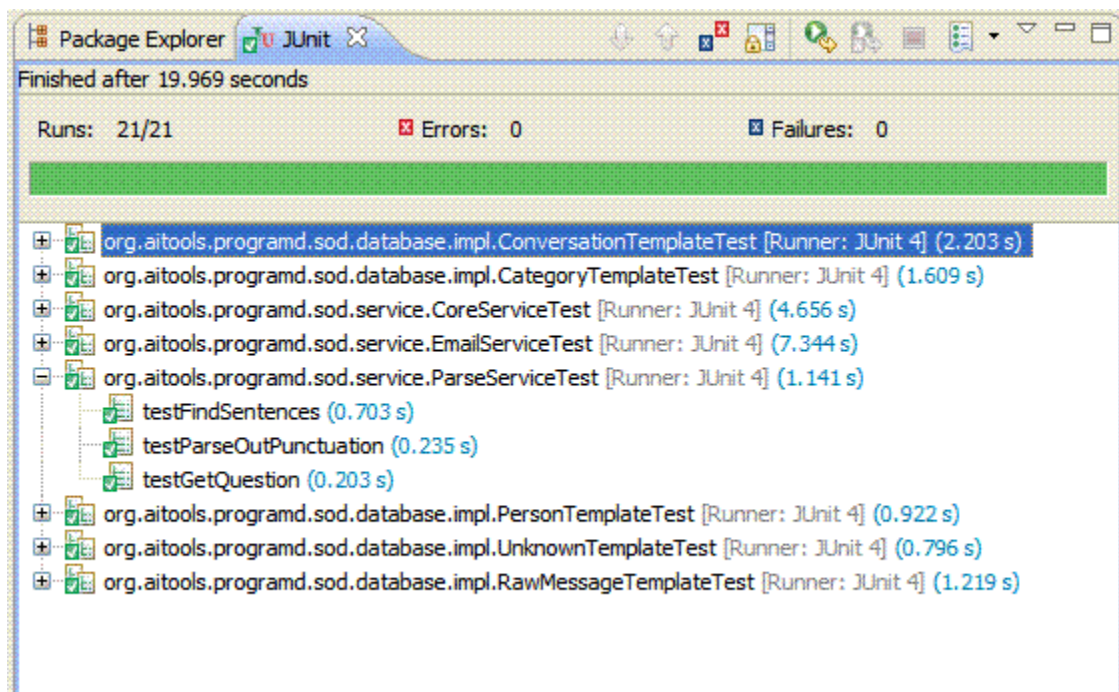
    String q = parseService.getQuestion(question);
    String nq = parseService.getQuestion(noQuestion);

    if (q != null && q.length() > 0 && nq == null) {
        assertTrue("Found the question, ignored the non-question.", true);
    }
    else {
        fail("That didn't work.");
    }
}

```

1.3.2.3 Test Execution

Eclipse allows all tests to be executed and shows the result of the test output. This feedback is shown below.



As can be seen, there are 21 JUnit tests that have been executed in under 20 seconds. This is a useful and efficient way to regularly verify the code base.

1.3.2.4 Test Configuration

The tests are configured in ServiceTest-context.xml file which can be found in the package under test/resources.

An example bean configuration is

```
<bean id="parseService"  
class="org.aitools.programd.sod.service.ParseService"/>
```

1.3.3 Spring

Where possible I have used the Spring Framework to simplify the development process. Spring is the most popular application development framework for enterprise Java*.

Spring facilitates the development of code that is

1. Easily tested
2. Reusable
3. Clearly structured

Specifically, Spring simplified database access for the application through the use of the JdbcTemplate object.

Spring objects were also used in the JUnit test scripts.

1.3.4 Ant

Apache Ant is a command line tool which builds a code base by following the instructions defined in the build file (build.xml). It is used in a similar fashion to 'make' in Unix.

It provides a robust and consistent build process. In order to start the build, the user should navigate to the project folder (c:\ProgramD) and call 'ant'.

```
C:\ProgramD>c:\apache-ant-1.8.2\bin\ant
```

The build.xml for this project is included in the Appendix.

1.3.5 ProgramD

ProgramD is the AIML platform that I have chosen to build into this application. According to its website[†], it is the most widely used open-source AIML bot platform.

* Source: www.springsource.org

1.3.6 AIML

AIML is an XML-compliant language which allows the user to configure automated responses from a bot. The important units of an AIML document are

<aiml> - tag that begins and ends the XML document.

<category> - tag that marks a unit of knowledge in the knowledge base.

<pattern> - tag that contains a simple pattern that the bot can match against.

<template> - tag that contains a detailed response to the pattern given above.

1.3.7 MySQL

1.3.7.1 Overview

MySQL is an open source database. It is reliable and easy to use and has a mature support structure which aides in the resolution of development issues.

1.3.7.2 Configuration Settings

The database settings are configured in c:\programd\conf\db.properties.

```
driver=com.mysql.jdbc.Driver
url=jdbc:mysql:///iehobbes
username=maureen
password=password
```

1.3.8 IMAP/SMTP

1.3.8.1 Overview

IMAP stands for internet message access protocol and is used to read incoming emails.

SMTP stands for simple mail transfer protocol and is used to send outgoing emails.

[†] www.aitools.org/Program_D

Gmail (www.gmail.com) is the host that is used to relay emails for this application.

1.3.8.2 Configuration Settings

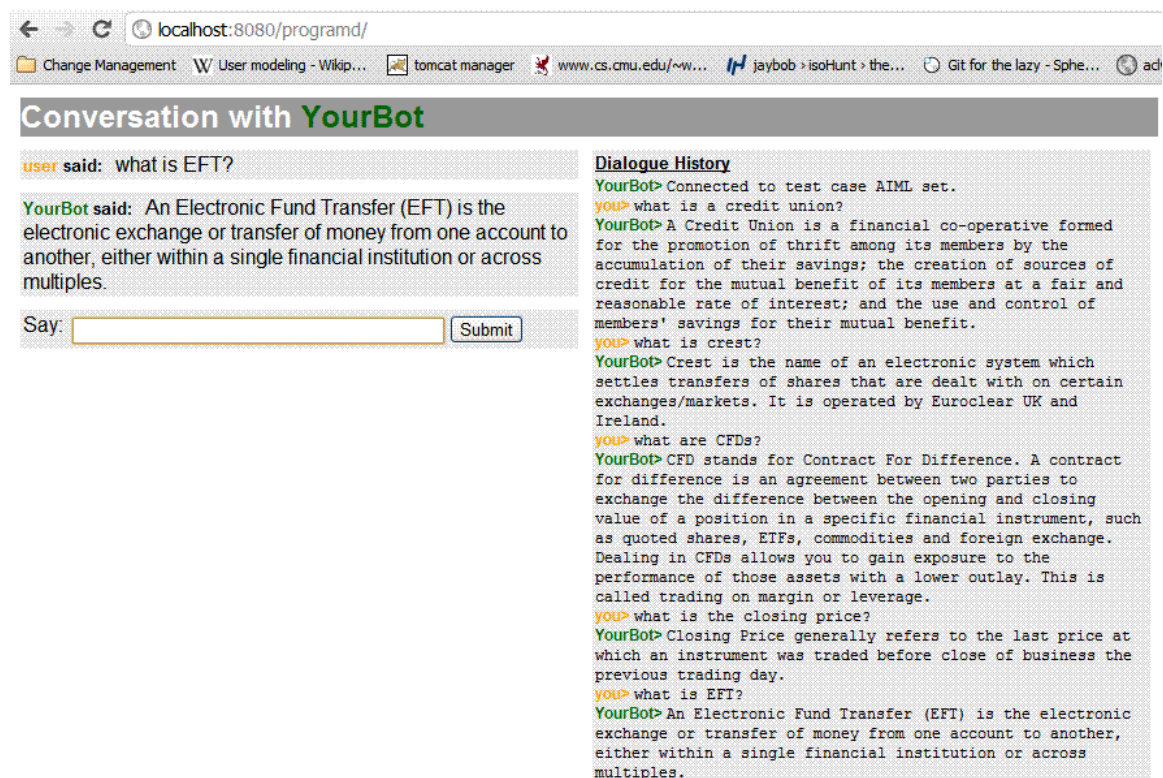
Email settings are configured in C:\ProgramD\conf\email.properties.

```
email=iehobb@iehobb@gmail.com
username=iehobb
password=colette.8
host=smtp.gmail.com
port=587
```

1.3.9 Tomcat

Tomcat is included as part of the ProgramD application. Here I have used it to test the generated AIML file.

The name ("YourBot") of the bot is specified in properties.xml under c:\programd\conf.

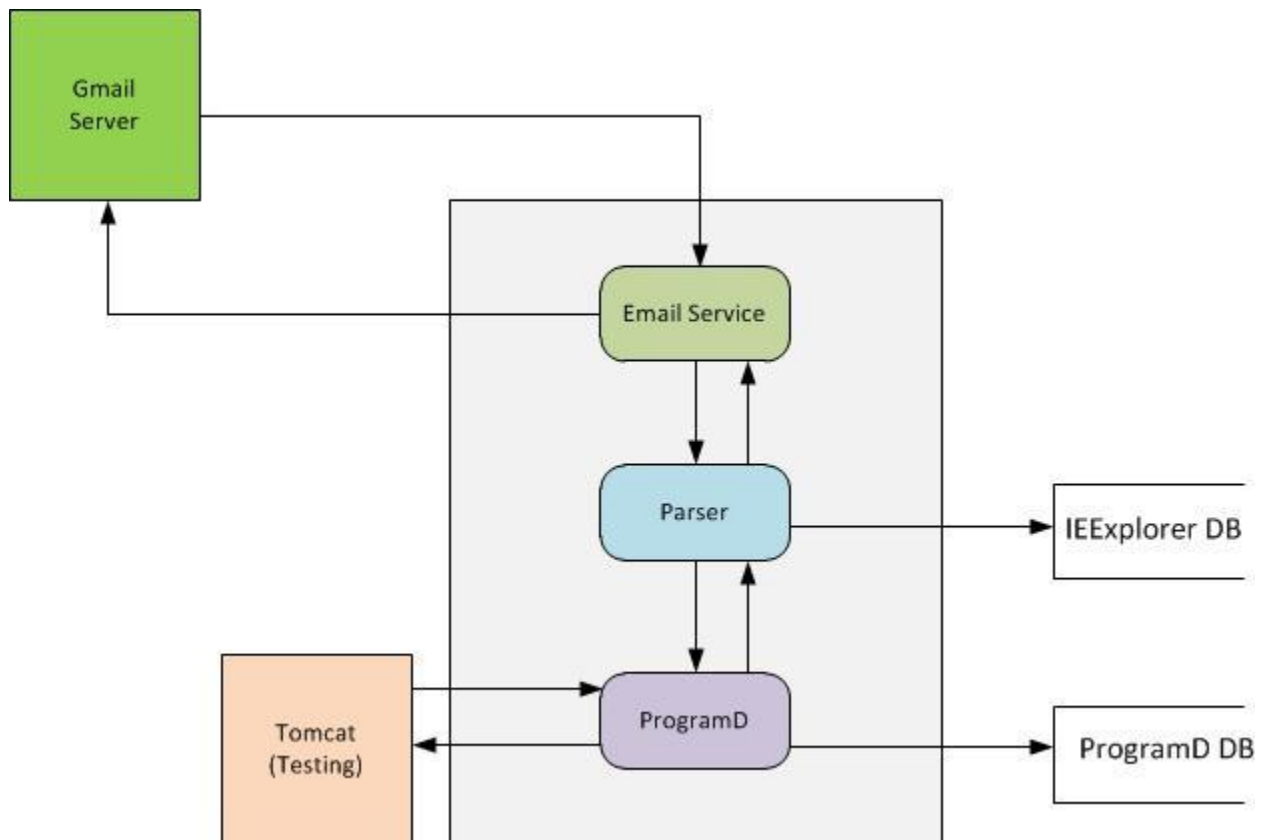


The screenshot shows a web browser window with the address bar displaying 'localhost:8080/programd/'. The browser's tab bar includes several open tabs: 'Change Management', 'User modeling - Wikip...', 'tomcat manager', 'www.cs.cmu.edu/~w...', 'jaybob > isoHunt > the...', 'Git for the lazy - Sphe...', and 'ad'. The main content area is titled 'Conversation with YourBot' and features a chat interface. On the left, the chat history is displayed, showing a user asking 'what is EFT?' and 'YourBot' responding with a definition of EFT. Below the chat history is a text input field with the placeholder 'Say:' and a 'Submit' button. On the right, a 'Dialogue History' section shows a list of recent chat messages, including 'YourBot' connected to the test case AIML set, and several user questions and 'YourBot' responses regarding credit unions, Crest, and CFDs.

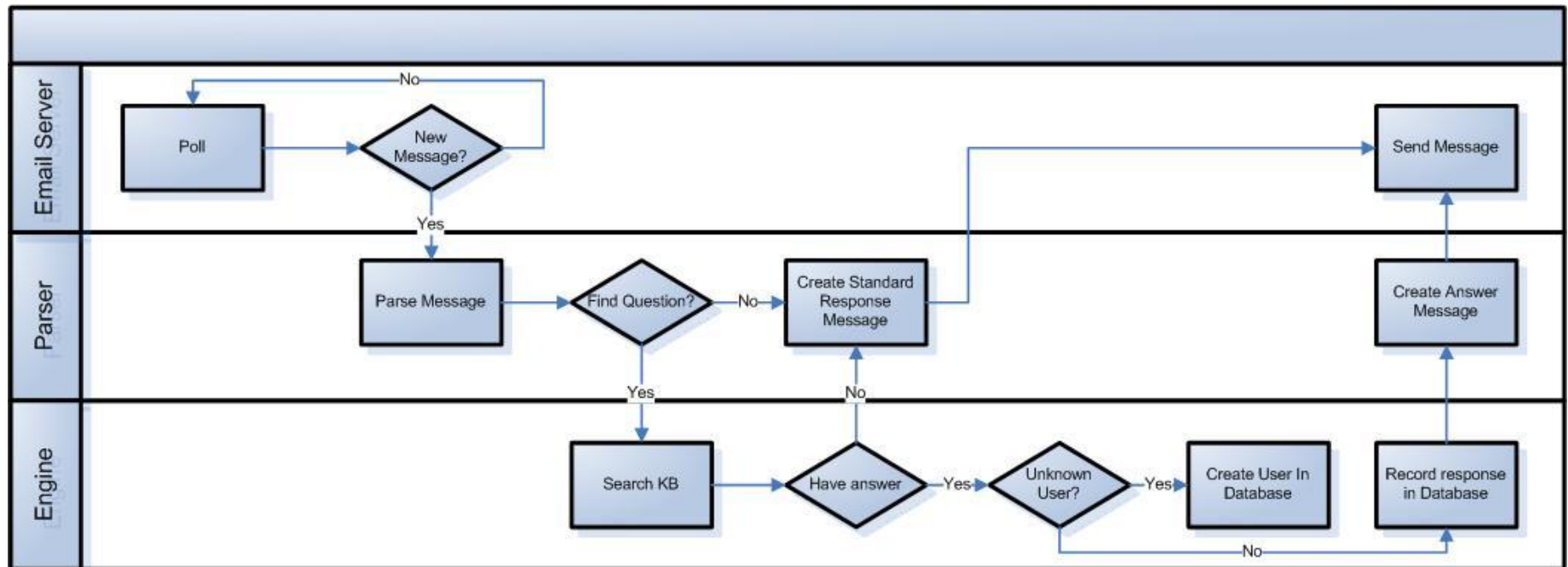
2 System

2.1 System Architecture

High Level System Architecture Diagram is shown below. All components except the Gmail server are installed on one server.



2.2 Process Flow Diagram



- Check if there's a new email
- If so, parse the message and find the questions
- Present the questions to the knowledge base engine
- Add the answers given to the message. Add any unknowns to the message
- If the user is unknown, create in the database
- Store the answers against the user
- Send the message to the user

2.3 Code base Structure

The application code base is built on the existing ProgramD code base and follows the package structure of same.

All code under the structure `org.aitools.programd.sod` was written by me.

All code in the Eclipse project was written by me.

All other code belongs to Noel Bush as the original owner of ProgramD.

All code is included on the CD. Only the code that I've written has been uploaded to moodle.

2.4 Requirements

2.4.1 Functional Requirements

The key functional requirements can be identified as

2.4.1.1 Email Exchange

Gmail provides a POP and IMAP interface which allows users to download messages from the Gmail servers without using the web GUI.

Equally, Gmail also provides support for sending email messages through an SMTP gateway.

As this is a fundamental requirement of the application, it is considered a high priority deliverable.

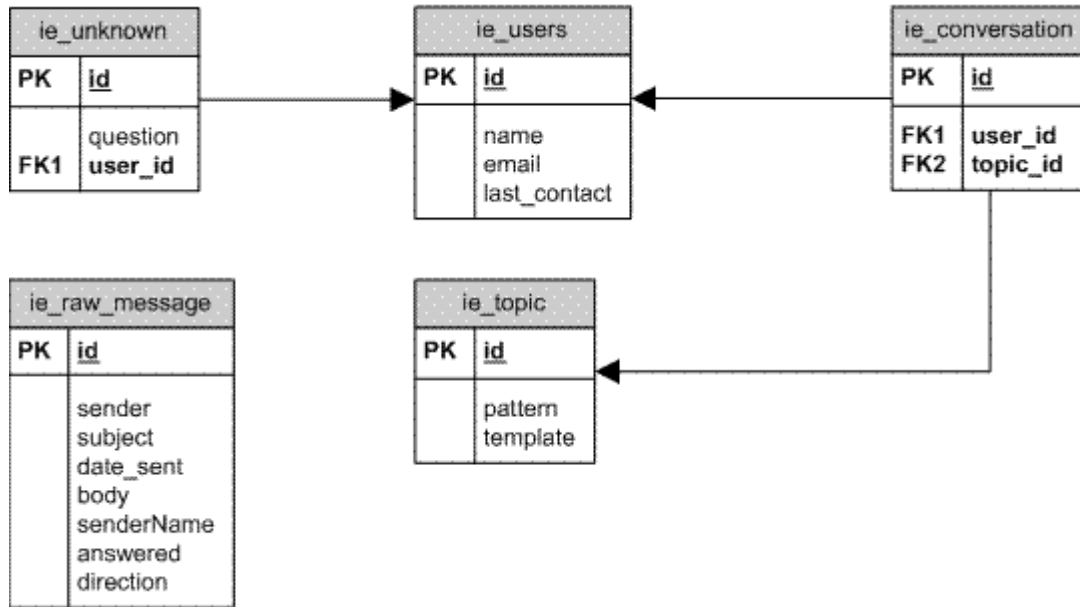
Risks & Technical Issues

There were no technical issues setting up email exchange.

As the application requires internet connectivity in order to function, there is a risk that it will not work if this is not available.

2.4.1.2 Database Access

In order to maintain a history of interactions with users, a database was required. The schema is shown below;



IE_USERS table stores the users who have interacted with the application.

IE_CONVERSATION stores the topics that a user has asked about.

IE_TOPIC stores the knowledge of the application.

IE_UNKNOWN stores the questions that there is no answer in the knowledge base for.

IE_RAW_MESSAGE stores the incoming and outgoing messages.

Risks & Technical Issues

There were no technical issues setting up the MySQL instance.

2.4.1.3 AIML Parser

One of the key requirements was the identification of an AIML parser.

Program D is the mostly widely used open source AIML bot platform[‡]. It has an open architecture which facilitated integration into my code base.

It is implemented in Java and I found it to be a robust AIML engine.

Risks & Technical Issues

[‡] From www.aitools.org

There is limited community activity. Any issues or problems need to be resolved in isolation.

2.4.2 Data Requirements

Data requirements for this application were focused on gathering and collating definitions and explanations for common and uncommon financial terms.

This information was obtained from many sources which are listed in the bibliography.

2.4.3 User Requirements

2.4.3.1 Generating AIML

In order to facilitate setting up the application, a program was written which transforms a Microsoft Excel spreadsheet into an AIML file.

The reason for this is that in most business situations, it is preferable that the business owner can manage the application themselves without constantly having to refer back to their IT resources.

The AIML is generated by running the following batch file;

```
C:\ProgramD\bin>aiml_gen.bat
```

The batch file starts the main class *IEAIMLGenerator.class*.

In a production environment, this would be setup as a scheduled task. The onus would rest with the business user to update the document before the scheduled task ran.

2.4.3.2 Running application

The main application program is also run from a batch file. The program is started by running the following;

```
C:\ProgramD\bin>ieexplorer-start.bat
```

This calls a batch file that starts the main class *IEExplorer.class*.

An obvious improvement to this process would be to configure the application as a windows service thereby negating the need to interactively log on in order to start the application.

2.4.4 Environmental Requirements

The application requires the following to run;

1. Internet access in order to send and receive emails

The application requires the following to build;

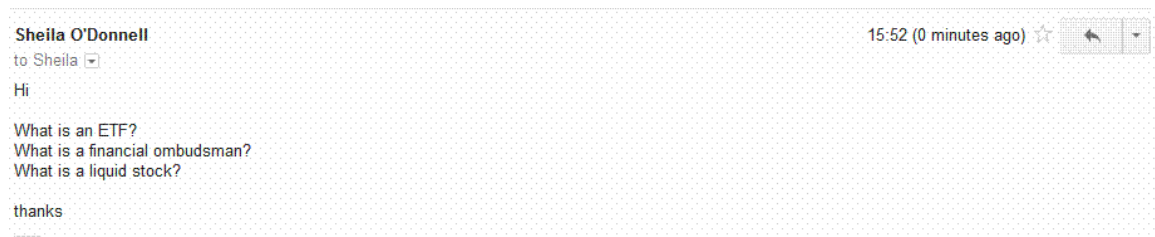
1. Apache Ant
2. Java 1.7

2.4.5 Usability Requirements

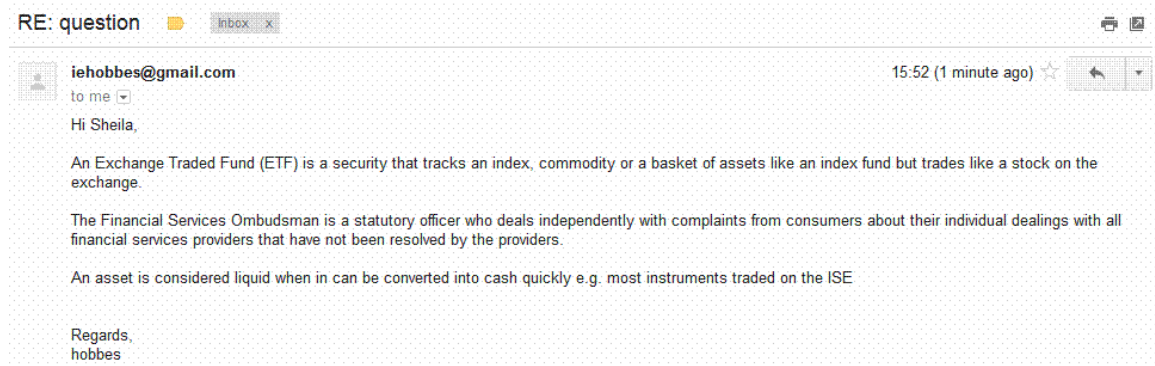
The user is expected to use email in order to interact with the application. It is assumed that the user is familiar with sending and receiving emails.

A sample email sent and received is shown below;

Sent:



Received:



2.5 Evaluation

2.5.1 Unit Testing

Unit testing ensures that the functionality of each core component is correct. For this development, I used JUnit to unit test as well as ad hoc testing. The JUnit tests are described earlier in this document (section 2.3.2).

Unit testing of the AIML code generated was also carried out on an ad hoc basis using the tomcat interface. This was quicker than waiting for an email response and easier to track that the correct answers were given.

2.5.2 System Testing

The purpose of the system testing is to ensure that all components work together correctly. The following system tests were carried out;

Test Name	Empty Email sent to Investment Explorer	
Before	Access to the internet	✓
	Access to an email client	✓
Steps	Create an email	✓
	Address to iehobbes@gmail.com	✓
	Leave subject blank	✓
	Leave body blank	✓
	Send	✓
Expected	Investment Explorer will respond but will ask the user to re-phrase their question	
Result	PASS	

Test Name	Email with one question	
Before	Access to the internet	✓
	Access to an email client	✓

Steps	Create an email Address to iehobbes@gmail.com Add a subject Add a question to the body Send	✓ ✓ ✓ ✓ ✓
Expected	1 answer returned	
Result	PASS	

Test Name	Email with many questions	
Before	Access to the internet Access to an email client	✓ ✓
Steps	Create an email Address to iehobbes@gmail.com Add a subject Add 3 questions to the body Send	✓ ✓ ✓ ✓ ✓
Expected	3 answers returned	
Result	PASS	

Test Name	Email with 1 unknown question	
Before	Access to the internet Access to an email client	✓ ✓
Steps	Create an email Address to iehobbes@gmail.com Add a subject Add 1 question that there is no answer in the KB Send	✓ ✓ ✓ ✓ ✓
Expected	Response will tell user that IE doesn't know	
Result	PASS	

Test Name	Email with many questions, 1 unknown	
Before	Access to the internet Access to an email client	✓ ✓
Steps	Create an email Address to iehobbes@gmail.com Add a subject Add 3 questions that are known and 1 unknown Send	✓ ✓ ✓ ✓ ✓
Expected	3 answers and 1 response saying that IE doesn't know	
Result	PASS	

3 Conclusions & Recommendations

3.1 *Application Improvements*

The parsing algorithm that is used in the application is very simple. A significant improvement to the application might be achieved by utilising a natural language programming package such as Apache OpenNLP (<http://opennlp.apache.org/>) to tokenise and parse the emails.

3.2 *Implementation Improvements*

It would be useful if the main class *IEExplorer.class* were installed as windows service. In that way, it wouldn't be necessary to logon interactively to start the process; it could be running whenever the server is available.

Unknown questions are currently stored in the database. It would be an improvement if the application regularly scanned this table and notified the admin user that these questions were missing rather than it being an action that must be driven by the user.

The AIML generator should accept CSV files as well as Microsoft Excel as this would negate the need to have Microsoft Excel installed. However as Excel is ubiquitous in a business environment, this improvement might be redundant.

4 Bibliography

Apache POI <http://poi.apache.org/>

Bush, N. Program D website. http://aitools.org/Main_Page

Davy Stockbrokers website. <http://www.davy.ie>

Irish Stock Exchange website. <http://www.ise.ie>

Flanagan, D. 2005. *Java in a Nutshell, 5th Edition*.

Git Community. <http://git-scm.com/>

Janetzko, D. 2008. *Assistance and Assessment in Tutorial Dialogues that Operate via E-mail*.

SpringSource Community. <http://www.springsource.org/>

U.S. Securities and Exchange Commission website.
<http://www.sec.gov/index.htm>

Walls, C. 2011. *Spring in Action*.

5 Appendix

5.1 Database create script

```
create table if not exists ie_users(  
    id INT not null auto_increment,  
    name varchar(100) default '',  
    email varchar(100) default '',  
    last_contact DATE,  
    PRIMARY KEY (id)  
);  
  
create table if not exists ie_raw_message(  
    id INT not null auto_increment,  
    sender varchar(100) default '',  
    subject varchar(500) default '',  
    date_sent date,  
    body varchar(5000) default '',  
    PRIMARY KEY (id)  
);  
  
create table if not exists ie_topic(  
    id INT not null auto_increment,  
    pattern varchar(250) default '',  
    template varchar(5000) default '',  
    PRIMARY KEY (id));  
  
create table if not exists ie_conversation(  
    id INT not null auto_increment,  
    user_id INT not null,  
    topic_id INT not null,  
    PRIMARY KEY (id));  
  
alter table ie_conversation add CONSTRAINT fk_ie_topic FOREIGN KEY  
(topic_id) references ie_topic (id);  
alter table ie_conversation add CONSTRAINT fk_ie_user FOREIGN KEY  
(user_id) references ie_users (id);  
  
create table if not exists ie_unknown(  
    id INT not null auto_increment,  
    question varchar(5000) default '',  
    user_id INT not null,  
    PRIMARY KEY (id));  
  
alter table ie_unknown add CONSTRAINT fk_ie_user_unknown FOREIGN KEY  
(user_id) references ie_users (id);
```

5.2 Ant build file

This is the ant build file that came with the ProgramD code base. I have edited to accommodate my new code.

```
<project default="jars" basedir=". ">
```

```

<!--Initialization Target-->
<target name="init">
    <tstamp />
    <property name="Name" value="Program D" />
    <property name="version" value="4.6" />
    <property name="year" value="2006" />

    <property name="javac.compiler" value="modern" />
    <property name="javac.debug" value="on" />
    <property name="javac.optimize" value="on" />
    <property name="javac.deprecation" value="on" />
    <property name="javac.jvm-target" value="1.5" />
    <property name="javac.source" value="1.5" />

    <property name="src.dir" value="${basedir}/JavaSource" />
    <property name="webapp.dir" value="${basedir}/WebContent" />
    <property name="webapp.lib.dir" value="${webapp.dir}/WEB-
INF/lib" />
    <property name="compile.lib.dir" value="${basedir}/lib" />
    <property name="distrib.dir" value="${basedir}/distrib" />
    <property name="spring.dir" value="C:/Program
Files/springsource/sts-2.8.1.RELEASE/plugins" />

    <property name="build.dir" value="${basedir}/build.tmp" />
    <path id="build.classpath">
        <pathelement location="${compile.lib.dir}/gnu.getopt-
1.0.10.jar" />
        <pathelement location="${compile.lib.dir}/jsp-api.jar" />
        <pathelement location="${webapp.lib.dir}/js.jar" />
        <pathelement location="${compile.lib.dir}/servlet-api.jar"
/>
        <pathelement location="${webapp.lib.dir}/log4j-1.2.13.jar"
/>
        <pathelement location="${webapp.lib.dir}/mail.jar" />
        <pathelement
location="${spring.dir}/org.springframework.jdbc_3.1.0.RC1.jar" />
    </path>
    <property name="javadoc.dir" value="${basedir}/docs/api" />

    <property name="main.jar" value="${distrib.dir}/programd-
main.jar" />
    <property name="rhino.jar" value="${distrib.dir}/programd-
rhino.jar" />

    <property name="war" value="${distrib.dir}/programd.war" />

    <property name="distrib.src.dir"
value="${distrib.dir}/distrib.src.tmp" />
    <property name="distrib.bin.dir"
value="${distrib.dir}/distrib.bin.tmp" />
    <property name="distrib.war.dir"
value="${distrib.dir}/distrib.war.tmp" />
    <property name="distrib.src.zip" value="${distrib.dir}/programd-
${version}-src.zip" />
    <property name="distrib.bin.zip" value="${distrib.dir}/programd-
${version}-bin.zip" />

```

```

        <property name="distrib.war.zip" value="${distrib.dir}/programd-
${version}-war.zip" />
        <property name="distrib.src.tar.bz2"
value="${distrib.dir}/programd-${version}-src.tar.bz2" />
        <property name="distrib.bin.tar.bz2"
value="${distrib.dir}/programd-${version}-bin.tar.bz2" />
        <property name="distrib.war.tar.bz2"
value="${distrib.dir}/programd-${version}-war.tar.bz2" />

        <filter token="year" value="${year}" />
        <filter token="version" value="${version}" />
        <filter token="date" value="${TODAY}" />
        <filter token="log" value="true" />
        <filter token="verbose" value="true" />

</target>

<!--Generate CoreSettings from core.xml.-->
<target name="core-settings" depends="init">
    <xslt in="conf/core.xml"
out="JavaSource/org/aitools/programd/CoreSettings.java"
style="resources/xslt/generate-settings-class.xslt">
        <factory name="net.sf.saxon.TransformerFactoryImpl" />
    </xslt>
</target>

<!--Prepare the build directory.-->
<target name="prepare" depends="init">
    <mkdir dir="${build.dir}" />
</target>

<!--Prepare the source code.-->
<target name="prepare-src" depends="init, prepare">
    <mkdir dir="${build.dir}" />
    <!--Copy resources.-->
    <copy todir="${build.dir}">
        <fileset dir="${webapp.dir}" includes="resources/schema/**"
/>
    </copy>
</target>

<!--Compile the source directory.-->
<target name="compile" depends="init, prepare-src">
    <javac srcdir="${src.dir}" destdir="${build.dir}"
classpathref="build.classpath" debug="${javac.debug}"
deprecation="${javac.deprecation}" optimize="${javac.optimize}"
target="${javac.jvm-target}" source="${javac.source}">
        <compilerarg value="-Xlint" />
        <compilerarg value="-Xlint:-path" />
        <compilerarg value="-Xlint:-serial" />
    </javac>
</target>

<!--Create the jars.-->
<target name="jars" depends="init, compile">
    <mkdir dir="${distrib.dir}" />
    <jar jarfile="${main.jar}">

```



```

        <fileset dir="${build.dir}/" includes="org/**"
excludes="org/aitools/programd/interpreter/RhinoInterpreter.class" />
        <fileset dir="${src.dir}/"
includes="org/aitools/programd/gui/icons/**" />
        <fileset dir="${build.dir}/" includes="resources/schema/**"
/>
    </jar>
    <jar jarfile="${rhino.jar}">
        <fileset dir="${build.dir}/"
includes="org/aitools/programd/interpreter/RhinoInterpreter.class" />
    </jar>
</target>

<!--Create a WAR file.-->
<target name="war" depends="init, compile">
    <delete file="${war}" />
    <war destfile="${war}" webxml="${webapp.dir}/WEB-INF/web.xml"
manifest="${webapp.dir}/META-INF/MANIFEST.MF">
        <fileset dir="${webapp.dir}">
            <include name="pages/**" />
        </fileset>
        <fileset dir="${basedir}">
            <include name="resources/**" />
            <exclude name="resources/database/**" />
            <exclude name="resources/icons/**" />
            <exclude name="resources/testing/**" />
            <exclude name="resources/xslt/**" />
        </fileset>
        <lib dir="${webapp.lib.dir}" />
        <classes dir="${build.dir}" />
        <classes dir="${basedir}/conf">
            <include name="log4j.xml" />
        </classes>
        <webinf dir="${webapp.dir}/WEB-INF">
            <include name="aiml.tld" />
            <include name="c.tld" />
            <include name="c-rt.tld" />
            <include name="dwr.xml" />
        </webinf>
    </war>
</target>

<!--Create the API documentation.-->
<target name="javadoc" depends="init, prepare-src">
    <mkdir dir="${javadoc.dir}" />
    <javadoc packageNames="*" sourcepath="${src.dir}"
destdir="${javadoc.dir}" classpathref="build.classpath" author="true"
version="true" use="true" splitindex="true" noindex="false"
windowtitle="${Name} API" doctitle="${Name} API">
        <link href="http://java.sun.com/j2se/1.5.0/docs/api/" />
        <link
href="http://www.urbanophile.com/arenn/hacking/getopt/" />
        <link href="http://java.sun.com/j2ee/1.4/docs/api/" />
    </javadoc>
</target>

<!--Create the schema documentation.-->

```

```

<target name="schema-doc" depends="init">
  <property name="xs3p" value="/usr/local/xs3p/xs3p.xsl" />
  <property name="schema" value="../resources/schema" />
  <property name="schema-doc" value="../docs/schema" />
  <delete dir="${schema-doc}" />
  <xslt in="${schema}/AIML.xsd" out="${schema-doc}/AIML.html"
style="${xs3p}">
    <param name="title" expression="A Schema for AIML 1.0.1" />
    <param name="sortByComponent" expression="false" />
  </xslt>
  <xslt in="${schema}/bot-configuration.xsd" out="${schema-
doc}/bot-configuration.html" style="${xs3p}">
    <param name="title" expression="Program D Bot Configuration
Schema" />
    <param name="sortByComponent" expression="false" />
  </xslt>
  <xslt in="${schema}/ChatLog.xsd" out="${schema-
doc}/ChatLog.html" style="${xs3p}">
    <param name="title" expression="Program D Chat Log Schema"
/>
    <param name="sortByComponent" expression="false" />
  </xslt>
  <xslt in="${schema}/test-cases.xsd" out="${schema-doc}/test-
cases.html" style="${xs3p}">
    <param name="title" expression="Program D Test Suite Schema"
/>
    <param name="sortByComponent" expression="false" />
  </xslt>
</target>

<!--Make the distribs.-->
<target name="distribs" depends="init, source-distribs, binary-
distribs, war-distribs" />

<!--The source distribs.-->
<target name="source-distribs" depends="init">
  <delete dir="${distrib.src.dir}" />
  <copy todir="${distrib.src.dir}">
    <fileset dir="${basedir}">
      <exclude name="*" />
      <exclude name="*/**" />
      <exclude name="aiml/**" />
      <exclude name="build.tmp/**" />
      <exclude name="classes/**" />
      <exclude name="distrib/**" />
      <exclude name="docs/**" />
      <exclude name="ffm/**" />
      <exclude name="lib/*-listener/*.jar" />
      <exclude name="logs/**" />
    </fileset>
  </copy>
  <zip destfile="${distrib.src.zip}">
    <zipfileset dir="${distrib.src.dir}" prefix="ProgramD/">
      <exclude name="bin/simple-console" />
      <exclude name="bin/simple-gui-console" />
    </zipfileset>
  </zip>
</target>

```

```

        <zipfileset dir="${distrib.src.dir}" prefix="ProgramD/"
filemode="744">
            <include name="bin/simple-console" />
            <include name="bin/simple-gui-console" />
        </zipfileset>
    </zip>
    <tar destfile="${distrib.src.tar.bz2}" compression="bzip2">
        <tarfileset dir="${distrib.src.dir}" prefix="ProgramD/">
            <exclude name="bin/simple-console" />
            <exclude name="bin/simple-gui-console" />
        </tarfileset>
        <tarfileset dir="${distrib.src.dir}" prefix="ProgramD/"
mode="744">
            <include name="bin/simple-console" />
            <include name="bin/simple-gui-console" />
        </tarfileset>
    </tar>
    <delete dir="${distrib.src.dir}" />
</target>

<!--The binary distribs.-->
<target name="binary-distribs" depends="init, jars">
    <delete dir="${distrib.bin.dir}" />
    <copy todir="${distrib.bin.dir}">
        <fileset dir="${basedir}">
            <include name="*" />
            <exclude name=".*" />
            <exclude name="*/**" />
            <exclude name="aiml/**" />
            <exclude name="build.tmp/**" />
            <exclude name="build.xml" />
            <exclude name="classes/**" />
            <exclude name="distrib/**" />
            <exclude name="docs/**" />
            <exclude name="ffm/**" />
            <exclude name="lib/*-listener/*.jar" />
            <exclude name="logs/**" />
            <exclude name="JavaSource/**" />
            <exclude name="secret.key" />
        </fileset>
        <fileset dir="${basedir}">
            <include name="distrib/programd-main.jar" />
            <include name="distrib/programd-rhino.jar" />
        </fileset>
    </copy>
    <zip destfile="${distrib.bin.zip}">
        <zipfileset dir="${distrib.bin.dir}" prefix="ProgramD/">
            <exclude name="bin/simple-console" />
            <exclude name="bin/simple-gui-console" />
        </zipfileset>
        <zipfileset dir="${distrib.bin.dir}" prefix="ProgramD/"
filemode="744">
            <include name="bin/simple-console" />
            <include name="bin/simple-gui-console" />
        </zipfileset>
    </zip>
    <tar destfile="${distrib.bin.tar.bz2}" compression="bzip2">

```

```

        <tarfileset dir="${distrib.bin.dir}" prefix="ProgramD/">
            <exclude name="bin/simple-console" />
            <exclude name="bin/simple-gui-console" />
        </tarfileset>
        <tarfileset dir="${distrib.bin.dir}" prefix="ProgramD/"
mode="744">
            <include name="bin/simple-console" />
            <include name="bin/simple-gui-console" />
        </tarfileset>
    </tar>
    <delete dir="${distrib.bin.dir}" />
</target>

<!--The war distribs.-->
<target name="war-distribs" depends="init, war">
    <delete dir="${distrib.war.dir}" />
    <copy todir="${distrib.war.dir}">
        <fileset dir="${basedir}">
            <include name="conf/**" />
            <include name="resources/testing/AIML.aiml" />
            <include name="resources/testing/AIML.xml" />
        </fileset>
        <fileset dir="${distrib.dir}">
            <include name="programd.war" />
        </fileset>
    </copy>
    <zip destfile="${distrib.war.zip}">
        <zipfileset dir="${distrib.war.dir}" prefix="ProgramD/" />
    </zip>
    <tar destfile="${distrib.war.tar.bz2}" compression="bzip2">
        <tarfileset dir="${distrib.war.dir}" prefix="ProgramD/" />
    </tar>
    <delete dir="${distrib.war.dir}"/>
</target>

<!--Clean up everything.-->
<target name="clean" depends="init, buildclean, distclean"/>

<!--Clean the build stuff.-->
<target name="buildclean" depends="init">
    <delete dir="${build.dir}"/>
    <delete file="${main.jar}"/>
    <delete file="${rhino.jar}"/>
</target>

<!--Clean up all the distrib stuff.-->
<target name="distclean" depends="init, source-distclean, binary-
distclean, war-distclean"/>

<!--Clean up the source distribs.-->
<target name="source-distclean" depends="init">
    <delete dir="${distrib.src.dir}"/>
    <delete file="${distrib.src.zip}"/>
    <delete file="${distrib.src.tar.bz2}"/>
</target>

<!--Clean up the binary distribs.-->

```

```

<target name="binary-distclean" depends="init">
    <delete dir="${build.dir}"/>
    <delete file="${main.jar}"/>
    <delete file="${rhino.jar}"/>
    <delete dir="${distrib.bin.dir}"/>
    <delete file="${distrib.bin.zip}"/>
    <delete file="${distrib.bin.tar.bz2}"/>
</target>

<!--Clean up the war distribs.-->
<target name="war-distclean" depends="init">
    <delete file="${war}"/>
    <delete dir="${distrib.war.dir}"/>
    <delete file="${distrib.war.zip}"/>
    <delete file="${distrib.war.tar.bz2}"/>
</target>

<!--Generate the multibot test files.-->
<target name="multibot" depends="init">
    <delete file="resources/testing/multibot/bots.xml"/>
    <xslt in="resources/xslt/generate-multibot.xslt"
        out="resources/testing/multibot/bots.xml"
        style="resources/xslt/generate-multibot.xslt">
        <factory name="net.sf.saxon.TransformerFactoryImpl"/>
    </xslt>
</target>
</project>

```

5.3 *ieexplorer-start.bat*

This is the main program batch file that runs the application - ieexplorer-start.bat.

```

@echo off

@rem Reset the quit variable.
set quit=

@rem Enter the bin directory.
pushd "%~p0"

@rem Check for needed environment space.
call common_functions.bat check_env %1 %2 %3 %4

@rem Get "base" directory (root of Program D installation)
if "%quit%"==" " call common_functions.bat set_base

@rem Configuration
set MAIN_CLASS=org.aitools.programd.sod.IEExplorer
set START_MEM=128m
set MAX_MEM=256m
set CORE_CONF=%BASE%\conf\core.xml

@rem Start Program D using the SimpleConsole main class.
if "%quit%"==" " call common_functions.bat start_programd %MAIN_CLASS% %START_MEM%
%MAX_MEM% "%CORE_CONF%"

:end
@rem On exit, go back to the original directory.
popd

```

This script depends on the common_functions.bat which is included in the build CD.

5.4 AIML generator bat

This is the AIML generator bat - aiml_gen.bat. It transforms the MS Excel file created by a business user into an AIML document that the application can consume.

```
@echo off

@rem Reset the quit variable.
set quit=

@rem Enter the bin directory.
pushd "%~p0"

@rem Configuration
set MAIN_CLASS=org.aitools.programd.sod.IEAIMLGenerator

@rem set the classpath
set CLASSPATH="c:\program files\springsource\sts-
2.8.1.RELEASE\plugins\org.springframework.jdbc_3.1.0.RC1.jar;C:\Program
Files\springsource\sts-
2.8.1.RELEASE\plugins\org.springframework.core_3.1.0.RC1.jar;C:\Program
Files\springsource\sts-
2.8.1.RELEASE\plugins\org.springframework.beans_3.1.0.RC1.jar;C:\Program
Files\springsource\sts-
2.8.1.RELEASE\plugins\org.springframework.transaction_3.1.0.RC1.jar;C:\Program
Files\springsource\sts-
2.8.1.RELEASE\plugins\org.springframework.asm_3.1.0.RC1.jar;C:\axis-1_4\lib\commons-
logging-1.0.4.jar;C:\ProgramD\lib\mysql-connector-java-3.1.12-
bin.jar;c:\programd\distrib\programd-main.jar;c:\programd\lib\poi-3.8-
20120326.jar;c:\programd\lib\poi-ooxml-3.8-20120326.jar"

@rem run the generator
if "%quit%"==" " java -cp %CLASSPATH% %MAIN_CLASS% "C:\Documents and Settings\Sheila\My
Documents\NCIRL\Dropbox\Project\Glossary.xls" C:\ProgramD\aiml\sod\glossary.aiml

@rem For tomcat testing, copy to
copy C:\ProgramD\aiml\sod\glossary.aiml C:\var\programd\resources\testing

:end
@rem On exit, go back to the original directory.
popd
```

5.5 AIML files

The AIML files generated are available in c:\programd\aiml\sod and are included on the CD.

5.6 Project Proposal

Project Proposal

Investment Explorer

Sheila O'Donnell, 11109815, sheila.m.odonnell@gmail.com

Degree Programme Name e.g. BSc (Hons) in Business Information
Systems

23/09/2011

Objectives

This project will build an expert system that will explore the difficulties of human/machine interaction. It will endeavour to overcome the typical problems that arise in these systems and will explore the following objectives ;

Explore internet application development with specific emphasis on SMTP/IMAP and the Twitter API

Understand the difficulties that exist in human/machine communication

Create a system that will interact 'intelligently' with a human counterpart. The aim will be to move towards a solution that is indistinguishable from a human response.

Utilise languages and technologies in a way to expand my understanding of their power and functionality

Background

Interaction between humans and machines has always presented problems typically due to the unpredictable way that humans communicate. Artificial Intelligence *is the science and engineering of making intelligent machines, especially intelligent computer programs. It is related to the similar task of using computers to understand human intelligence*[§]

This project will build an AIML system with a developed knowledge base that will interact with humans via email and twitter. The content of the knowledge base will likely be an expert on products available to retail investors^{**} in Ireland. The system will attempt to answer typical questions about the availability of assets, the risks associated with same and the likely return on investment.

The system will attempt to be aware of its own limitations and forward the question to a human if it is beyond its knowledge base.

Technical Approach

The technical approach will be as follows;

Research & Design

Research

Requirements Analysis – document clearly the deliverables.

Testing/Validation Plan – document an ongoing testing/validation plan (system test) as well as an end-user testing approach

[§] John McCarthy, Stanford University 2007 (<http://www-formal.stanford.edu/jmc/whatisai/>)

^{**} Retail investor is one who buys and sells assets for his/her own portfolio. They are unlikely to have professional investment experience

Implementation Plan – it may be necessary to host the system in a location other than the development server, in which case an implementation plan will be required

Prototype – this early prototype will confirm basic functionality but will be missing the richness of a fully developed AIML knowledge base

Review & enhancement – following a technical review of the prototype, improvements will be identified and applied

Prototype – this prototype will contain most of the deliverable functionality but will require some further enhancements

Review & finish – no major changes are expected at this point

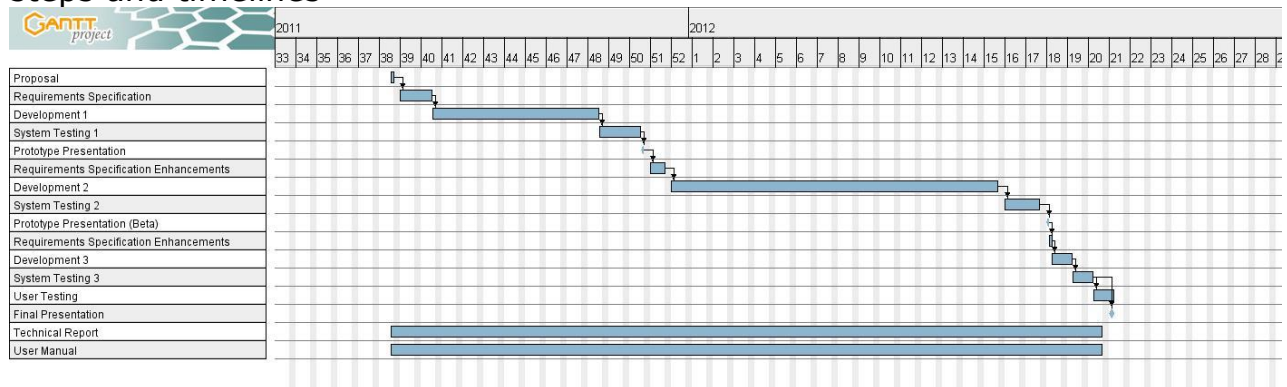
Final presentation

Special resources required

None at this time

Project Plan

Gantt chart using Microsoft Project with details on implementation steps and timelines



Technical Details

Key to the success of the system will be the design of the AIML knowledge base. It is expected that the design will cover the following functionality

1. Processing questions – the system will accept input via email and/or twitter. This will require that the system can parse the content from the messages so that it can be analysed.
2. Answering questions – the system will use the knowledge base to find a suitable answer for the user and then wrap the response in the correct protocol.
3. Maintain conversations – the system will be aware of conversations it has had from the same source and will be able to refer back to information already shared with this user.

4. Clarifying – the system will be able to challenge the users request if it is unclear with a view to responding well
5. Limitations – the system will recognise if the information is not stored in its knowledge base and respond to the user in kind. It will also notify the system administrator that the knowledge base is lacking information and needs to be addressed.

It is expected that the project will utilise the following technologies

- AIML – an XML compliant language that will be used to define the knowledge base
- Program E or similar
- MySQL – database that will be used
- Apache Webserver/Tomcat Application Server
- Java will be the primary development language though PHP may also be used
- SMTP and IMAP protocols – using Gmail as it provides an IMAP interface for incoming email and SMTP protocol for outgoing emails. This will negate the need to setup a separate email server^{††}.
- Twitter API

System Overview

The system will consist of

Server which will host the AIML knowledge base and the automated response agent.

The agent will scan the system inbox for messages and the twitter feed for questions and will respond using the same medium.

The agent will attempt to distinguish between genuine messages and 'junk'.

The agent will notify the system administrator when the knowledge base is lacking

Evaluation

The project will be evaluated by 2 methods;

Technical Assessment – there will be a number of tests designed to confirm that the system responds meaningfully in the majority of cases. Errors (responding with a standard answer or fail to respond) will be used as a yardstick to gauge improvement over time.

Human Trials – it will be necessary to ask for human interaction to show that the system is answering in a useful way. In order to control the human interaction and to obtain quantifiable results, a questionnaire will be designed to guide testers.

^{††} ICANDO project 2008, School of Computing, National College of Ireland

Consultation 1

Dr Dietmar Janetzko

"Leave out objective #2 and #4. They emphasize your personal development. Clearly, this is essential. However, it common practice to emphasize engineering-related objectives."

"I will connect you with another student who also addresses this project so that join collaborate on the common core while also specialising on the particular aspects of each of you individual projects."

"... a deliverable of a previous project that provides some insight into this type of application. Please consider in particular section 2.9.3"^{##}

Consultation 2

Dr Stephan Weibelzahl

"One thing that you might want to clarify further is the process of requirements/information elicitation. It would be great if you had a systematic approach to creating the database. I can see aspects of this (e.g., the different types of questions you anticipate), but how do you find out what possible questions are? How do you figure out what the answer is?"

"The proposed feature where the system would recognise its own limitations is an interesting one. AIML has some built in features to do this (e.g., no match), but maybe you can explore this in more depth, as it is a typical shortcoming of expert systems."

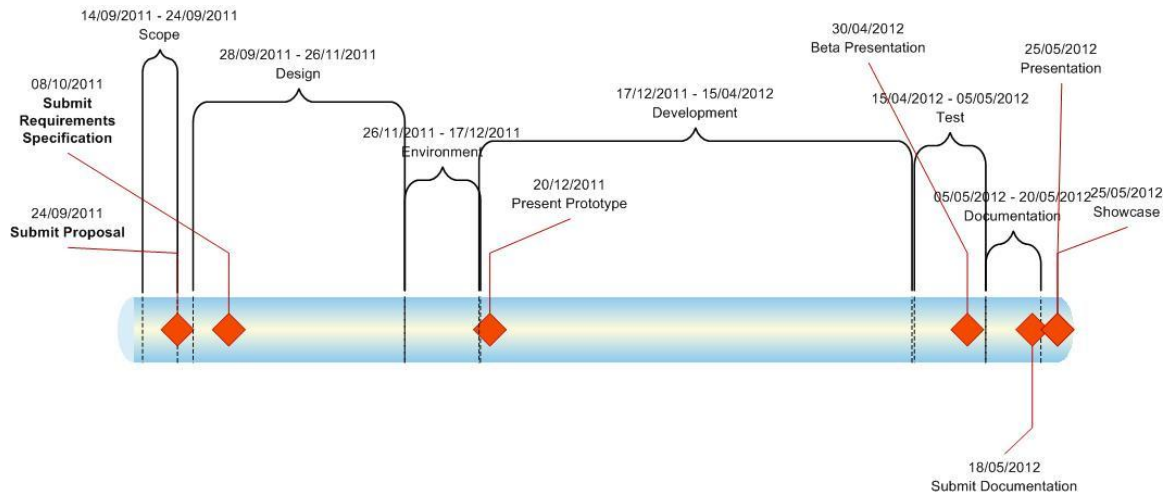
Proposed Supervisor

Names of academic staff member that has agreed to act as a supervisor for this project.

Signature of student and date

5.7 Project Plan

^{##} ICANDO project 2008, School of Computing, National College of Ireland



5.8 Requirements Specification

Title Requirements Specification (RS)

Document Control

Revision History

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
04/10/11	Draft	Create	SOD	X	X

Distribution List

Name	Title	Version
Paul Stynes	Lecturer II	2
Dr Dietmar Janetzko		

Related Documents

Title	Comments
Title of Use Case Model	
Title of Use Case Description	

Table of Contents

Title Requirements Specification (RS)	36
Document Control	36
Revision History	36
Distribution List	36
Related Documents	36
1 Introduction	39
1.1 Purpose	39
1.2 Project Scope	39
1.3 Definitions, Acronyms, and Abbreviations	39
2 User requirements definition	39
3 System architecture	40
4 Requirements specification	41
4.1 Physical environment requirements	42
4.1.1 Requirement 1 'Investment Explorer' Server	42
4.1.2 Requirement 2 Email Server	42
4.2 Interface requirements	43
4.2.1 Requirement 1 Parse Message	43
4.2.2 Requirement 2 Parse Inbox	43
4.2.3 Requirement 2 Consume IMAP message	43
4.2.4 Requirement 3 Generate SMTP message	44
4.2.5 Requirement 4 Identification of major topics in an email	44
4.2.6 Requirement 5 Junk Filter	44
4.3 Functional requirements	45
4.3.1 Requirement 1 - <Bot> Answer Question	45
4.3.2 Requirement 2 - <Bot> Ask for Clarification	47
4.3.3 Requirement 3 - <Bot> Prompt User for Further Information	48
4.3.4 Requirement 4 – Recognition of Knowledge Limitation	49
4.3.5 Requirement 5 – Spawn email to Human Expert	50
4.3.6 Requirement 6 – Notify Admin of missing Knowledge	51
4.3.7 Requirement 7 – User Modeling of a potentially large number of senders	52
4.4 Documentation requirements	54
4.4.1 Requirement 1 User Manual Description & Priority	54
4.4.2 Requirement 2 System Configuration Description & Priority	54
4.5 Data requirements	55
4.5.1 Requirement 1 General Knowledge Data Description & Priority	55
4.5.2 Requirement 2 Investment Knowledge Data Description & Priority	55
4.5.3 Requirement 3 Catch-all Data	55

Description & Priority	55
4.5.4 Requirement 4 User Model Data	56
Description & Priority	56
Non-Functional Requirements	57
4.5.5 Performance/Response time requirement	57
4.5.6 Availability requirement	57
4.5.7 Recover requirement	57
4.5.8 Robustness requirement	57
4.5.9 Security requirement	57
4.5.10 Reliability requirement	57
4.5.11 Maintainability requirement	57
4.5.12 Portability requirement	57
4.5.13 Extendibility requirement	57
4.5.14 Reusability requirement	57
4.5.15 Resource utilization requirement	58
5 System models	58
6 System evolution	58
7 Appendices	59
Use case 4.3.1	59
Use case 4.3.2	60
Use case 4.3.3	61
Use case 4.3.4	62
Use case 4.3.5	64
Use case 4.3.6	65
Use case 4.3.7.6.1	66
Use case 4.3.7.6.2	67
Use case 4.3.7.6.3	69

6 Introduction

6.1 Purpose

The purpose of this document is to set out the requirements for the development of an expert system that will answer user questions about investment products.

The intended customers are people who need to make a decision about their approach to their financial future.

6.2 Project Scope

The scope of the project is to develop a system that will allow users to access an 'Expert' in investment products with a view to choosing one that best suits their financial objectives.

The system will have

- an interface via Email
- a developed AIML knowledge base
- a system defined expert to answer the questions

6.3 Definitions, Acronyms, and Abbreviations

AIML – Artificial Intelligence Markup Language

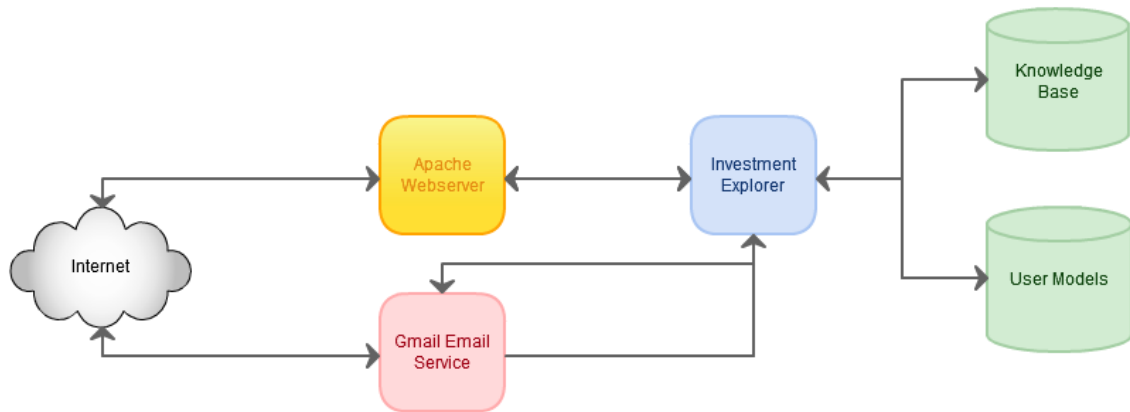
KB – Knowledge Base

MiFID – Markets in Financial Instruments Directive

7 User requirements definition

This section is not applicable to this document.

8 System architecture



[online diagramming & design] creately.com

The primary method of communication will the system will be via the Gmail email service. The Apache webserver will be used for testing purposes.

9 Requirements specification

All requirements should be verifiable. For example, experienced controllers shall be able to use all the system functions after a total of two hours training. After this training, the average number of errors made by experienced users shall not exceed two per day.

The following table demonstrates how to make requirements verifiable incorporate metrics as follows:

Property	Measure	Reasoning
Speed	Respond in 30 minutes under	Communication is asynchronous therefore immediate response will not be measured. The system will however be expect to respond in a timely fashion.
Size	Unknown yet	At design, it is difficult to determine the size requirement of the system
Ease of Use	1 - 2 hours learning	The UI is email. As this medium is in common usage, it is expected that users will not have any difficulty using the system
Reliability		
Robustness		
Portability	None	User will be unaware of the platform on which the system is operating

9.1 *Physical environment requirements*

This section describes the environment where the equipment is to function and if there are any environmental restrictions, such as temperature, humidity or magnetic interference.

9.1.1 Requirement 1 'Investment Explorer' Server

The system requires a server on which to operate. This server will house the email client, database and expert functionality of the system.

It will also host an Apache webserver which will allow the AIML knowledge base to be tested in isolation without utilising the email server.

9.1.2 Requirement 2 Email Server

Googles Gmail will be used as the email service for this system. The reasoning behind this is that Google provide an IMAP interface to read emails and an SMTP interface to send them.

This negates the need to setup an independent email server and eliminates the complication of managing such a server.

9.2 Interface requirements

9.2.1 Requirement 1 Parse Message

9.2.1.1 Description & Priority

The system will be required to parse an email message into the follow parts;

1. Message headers – this will be discarded
2. Email address – to uniquely identify the user
3. Topic(s) – these will be presented to the system as questions

Priority

The priority of this requirement is HIGH.

9.2.2 Requirement 2 Parse Inbox

9.2.2.1 Description & Priority

The system will be required to parse an email inbox where the user messages are stored.

It will be necessary to perform the following tasks

4. Login to the mail server
5. Logout of the mail server
6. Read message
7. Move message
8. Delete message

Priority

The priority of this requirement is HIGH.

9.2.3 Requirement 2 Consume IMAP message

9.2.3.1 Description & Priority

Googles Gmail service will be used to handle incoming and outgoing email messages thereby negating the need to setup a dedicated email server.

The message will be passed to 'Parse Message' requirement.

Priority

The priority of this requirement is HIGH.

9.2.4 Requirement 3 Generate SMTP message

9.2.4.1 Description & Priority

The answer returned from the will be packaged as an SMTP message and send to the mail server.

Priority

The priority of this requirement is HIGH.

9.2.5 Requirement 4 Identification of major topics in an email

9.2.5.1 Description & Priority

In order to present the question to the system, it will necessary to parse the question from the email.

There may be more than one question in the email.

Priority

The priority of finding one question is HIGH.

The priority of finding subsequent questions in the same message is MEDIUM.

9.2.6 Requirement 5 Junk Filter

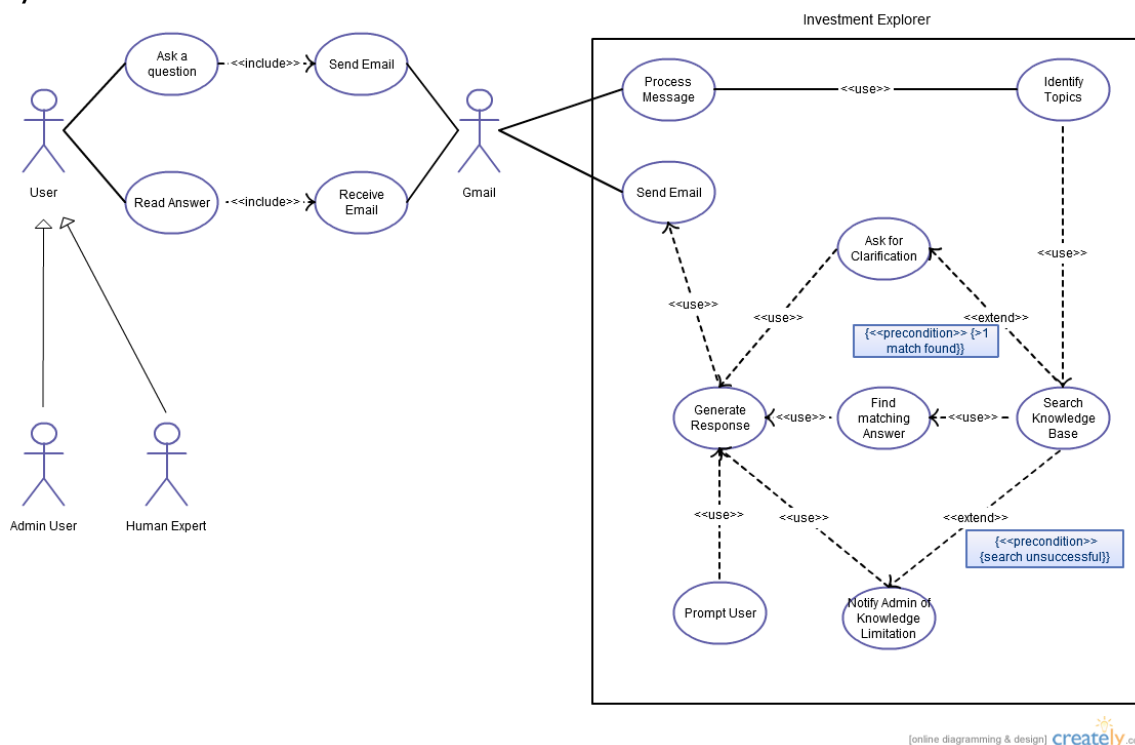
9.2.6.1 Description & Priority

From time to time, messages are inadvertently sent to the wrong recipient. This requirement will ensure that these messages are not treated as actual input for the system.

Priority

The priority of this requirement is MEDIUM.

This section lists the functional requirements which describe the system shown below.



9.3.1 Requirement 1 - <Bot> Answer Question

9.3.1.1 Description & Priority

The system will present the question to the automated response agent (bot) which will in turn search the AIML Knowledge base for an answer.

If an answer is found, it will be returned to the calling object.

The priority of this requirement is HIGH.

9.3.1.2 Requirement Activation

The user has no direct access to this requirement. The parsing of the messages sent by the user will be handled by another requirement and the question presented to this requirement.

9.3.1.3 Technical issues

No technical issues with the requirement at this time.

9.3.1.4 Risks

This functionality is key to the success of the system.

9.3.1.5 Dependencies with other requirements

This requirement depends on interface requirement 1 – Parse Message.

9.3.1.6 Functional Requirements

This requirement is described in Use case 4.3.1

9.3.2 Requirement 2 - <Bot> Ask for Clarification

9.3.2.1 Description & Priority

This requirement builds on requirement 1. If the answer that is found by the agent is ambiguous, the agent will request a clarification from the user.

The priority of this requirement is MEDIUM.

9.3.2.2 Requirement Activation

This requirement will only be called when there is more than one answer that could match the question.

9.3.2.3 Technical issues

There are no perceived technical issues with this requirement at this time.

9.3.2.4 Risks

This requirement may still fail if

- The clarification returned by the user does not dismiss the ambiguity. The agent will continue to question the user in order to improve the answer.
- The user may deliberately or inadvertently answer nonsense.

9.3.2.5 Dependencies with other requirements

This requirement depends on requirement 1.

9.3.2.6 Functional Requirements

This requirement is described in Use case 4.3.2

9.3.3 Requirement 3 - <Bot> Prompt User for Further Information

9.3.3.1 Description & Priority

It may be necessary for the agent to prompt the user for further information. This might occur if the agent has asked for a clarification but not yet received a response from the user. The agent will wait an acceptable amount of time before prompting. An acceptable amount of time will need to be defined. The agent will only prompt once – it may be the case that the user has found the answer elsewhere. The priority of this requirement is LOW.

9.3.3.2 Requirement Activation

The user will not trigger this requirement. This requirement will only be called if the conversation has lulled i.e. the agent has sent a message to the user and is expecting a response.

9.3.3.3 Technical issues

It will be difficult to determine an acceptable time if the user is finished questioning the system. The user is under no obligation to notify the system that he is finished.

9.3.3.4 Risks

This requirement has the potential to send too many unsolicited messages to the user therefore it will be important to restrict its use.

There will be a configuration setting that will allow the Admin user to turn this functionality off.

9.3.3.5 Dependencies with other requirements

This requirement depends on requirement 1 and 2.

9.3.3.6 Functional Requirements

This requirement is described in Use case 4.3.3

9.3.4 Requirement 4 – Recognition of Knowledge Limitation

9.3.4.1 Description & Priority

If the system is unable to answer the question asked, it will do the following;

- Inform the user
- Allow the user to escalate to a human expert
- Notify the admin user that there is a gap in the AIML knowledge base

The priority of this requirement is MEDIUM.

9.3.4.2 Requirement Activation

The user will not access this requirement directly. This functionality will be required if the system is unable to find an answer.

9.3.4.3 Technical issues

The success of the system will be measured by how infrequently this functionality is required.

9.3.4.4 Risks

In order to simulate a human experience, the system needs to develop a rich knowledge base. Limitations of the existing KB must be recognized in order to enhance it.

9.3.4.5 Dependencies with other requirements

This requirement depends on the interface requirements "Generate SMTP message" and "Consume SMTP message"

9.3.4.6 Functional Requirements

This requirement is described in Use case 4.3.4

9.3.5 Requirement 5 – Spawn email to Human Expert

9.3.5.1 Description & Priority

If the knowledge base does not contain the answer to a question, the system will spawn an email to a human expert. For the purposes of this project, the human expert and the admin user are the same.

This requirement will be called in tandem with requirement 6.

Once the system has sent the email, the responsibility of responding to the user transfers to the human expert.

The priority of this requirement is MEDIUM.

9.3.5.2 Requirement Activation

The user will be unaware of this functionality. It will appear to the user that another expert has been drafted in to help with their question.

9.3.5.3 Technical issues

No technical issues identified at this time.

9.3.5.4 Risks

No risks identified at this time.

9.3.5.5 Dependencies with other requirements

This requirement will be called in tandem with requirement 6 and is dependent on requirement 4.

9.3.5.6 Functional Requirements

This requirement is described in Use case 4.3.5

9.3.6 Requirement 6 – Notify Admin of missing Knowledge

9.3.6.1 Description & Priority

If the knowledge base does not contain the answer to a question, the system will notify the admin user that there is a gap in the knowledge base. For the purposes of this project, the human expert and the admin user are the same.

This requirement will be called in tandem with requirement 5. Once the system has sent the email, it is the responsibility of the admin user to update the knowledge base so as to improve the responses of the system.

The priority of this requirement is MEDIUM.

9.3.6.2 Requirement Activation

The user will be unaware of this functionality.

9.3.6.3 Technical issues

No technical issues identified at this time.

9.3.6.4 Risks

It is possible that the question may be beyond the knowledge of the human expert. At this time, there is no mitigation for this risk.

9.3.6.5 Dependencies with other requirements

This requirement will be called in tandem with requirement 5 and is dependent on requirement 4.

9.3.6.6 Functional Requirements

This requirement is described in Use case 4.3.6

9.3.7 Requirement 7 – User Modeling of a potentially large number of senders

9.3.7.1 Description & Priority

It is important that the system is able to maintain a conversation with an interested user. In order to do so, it will be necessary to use a method of user modelling.

In order to build a user model, it will be necessary to question the user in order to establish the following;

- Personal profile – age, education
- Attitude to risk
- Time horizon
- Objectives/Goals

In this way, the system can respond with an answer that is tailored to the user.

Over time, the model will become richer and allow the system to respond with greater specificity.

The priority of this requirement is HIGH.

9.3.7.2 Requirement Activation

Initial questions will be answered generically but a record will be kept of questions asked by a returning user. When the user returns to the system many times, the system will seek to establish the user model by questioning the user.

9.3.7.3 Technical issues

Although User Modeling is a mature discipline, it is new to the developer. It is expected that significant time will be required to research and develop this functionality.

9.3.7.4 Risks

This is a new area of learning for the developer therefore there is a risk that this deliverable may not be meant. This risk can be mitigated by the following;

1. Research into the discipline
2. Prototyping the functionality early to discover gaps

9.3.7.5 Dependencies with other requirements

This requirement does not depend on other requirements though it will add a richness to the system overall.

9.3.7.6 Functional Requirements

9.3.7.6.1 *New user*

This requirement is described in Use case 4.3.7.6.1

9.3.7.6.2 *Returning user*

This requirement is described in Use case 4.3.7.6.2

9.3.7.6.3 *Establish knowledge about user*

This requirement is described in Use case 4.3.7.6.3

9.4 Documentation requirements

Examples of document requirements are

- How much documentation is required
- Should it be on-line, in book format or both
- To what audience is each type of documentation addressed

9.4.1 Requirement 1 User Manual

Description & Priority

The user manual will be a short document that will describe to a novice user how to use the 'Investment Explorer'. It will be sent to the user when he or she requests help from the system.

Priority

The priority of this requirement is LOW.

9.4.2 Requirement 2 System Configuration

Description & Priority

The system configuration document will cover the following topics;

- System installation and setup
- Adding additional information to the AIML Knowledge Base
- Adding additional filters to the junk mail filters

The target audience for this document is an IT professional who is a competent administrator of Windows based systems.

Priority

The priority of this requirement is MEDIUM.

9.5 Data requirements

Data requirements should consider the following:

- What should the format of data be for input and output
- How often will they be sent or received
- How accurate must they be
- To what degree of precision must the calculations be made
- How much data flow through the system
- Must the data be retained for any period of time

9.5.1 Requirement 1 General Knowledge Data

Description & Priority

Although the systems main function is to answer questions about investment products, it will be able to hold a generic conversation with the user.

Priority

The priority of this requirement is LOW.

9.5.2 Requirement 2 Investment Knowledge Data

Description & Priority

The Knowledge Base will be built using the following data sources;

- Information published by National Consumer Agency
- Information published by the Central Bank of Ireland (Financial Services Authority)
- Information published by the Irish Stock Exchange
- Retail products offered by financial services companies in Ireland

Priority

The priority of this requirement is HIGH.

9.5.3 Requirement 3 Catch-all Data

Description & Priority

When the system is unable to find an answer for the user, it will still be necessary that it responds meaningfully. The purpose of this data is

- at best to respond in kind (in reference to the topic) or

- at worst to respond generically.

Priority

The priority of this requirement is MEDIUM.

9.5.4 Requirement 4 User Model Data**Description & Priority**

The system requires a data store for the User Modeling information. This will allow the system to maintain a conversation with the user.

Priority

The priority of this requirement is MEDIUM.

Non-Functional Requirements

Specifies any other particular non-functional attributes required by the system. Examples are provided below. Remove the requirement headings that are not appropriate to your project.

9.5.5 Performance/Response time requirement

The communication between system and user is asynchronous. However, the system will respond to the user in less than 30 minutes.

9.5.6 Availability requirement

For the purposes of this project, the system will only be available when the processes are started by the admin user.

9.5.7 Recover requirement

See Availability requirement.

9.5.8 Robustness requirement

See Availability requirement.

9.5.9 Security requirement

The system will interface via email therefore it will be necessary to ensure that the user modelling aspect is effective. If the system is unsure of the identity of the user, it will assume that this is a new user.

9.5.10 Reliability requirement

See Availability requirement

9.5.11 Maintainability requirement

From time to time the regulations applied to investment products change (e.g. MiFID 2004). When these occur, the system will be updated before the directive comes into effect.

9.5.12 Portability requirement

Not applicable to this development.

9.5.13 Extendibility requirement

Not applicable to this development.

9.5.14 Reusability requirement

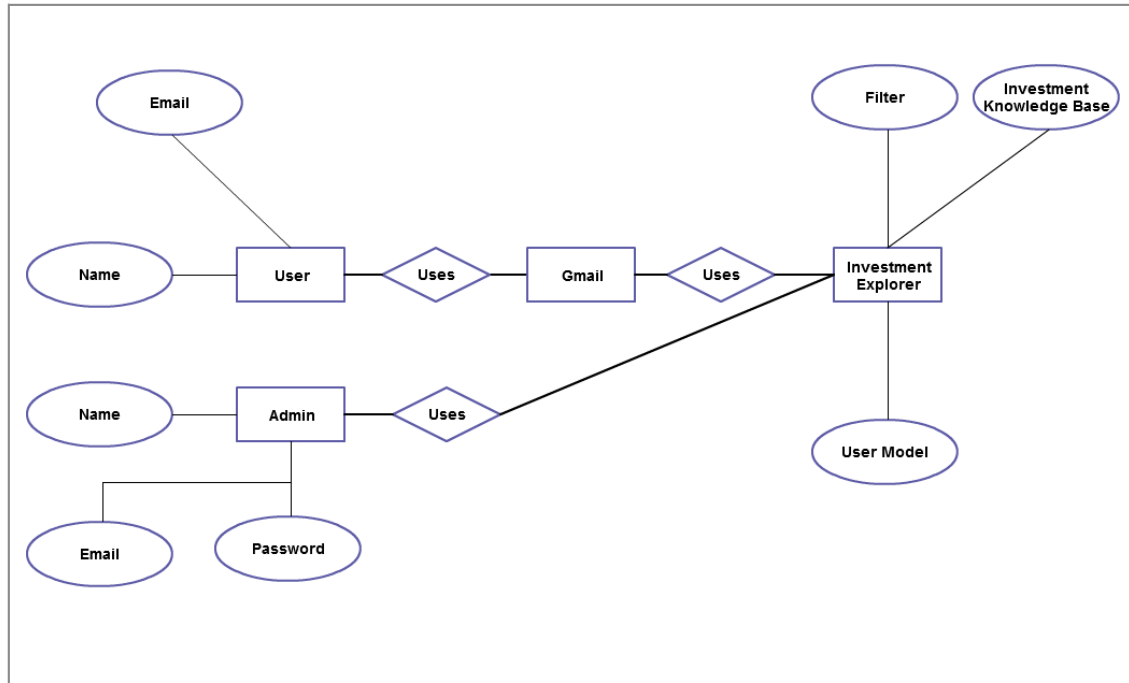
Not applicable to this development.

9.5.15 Resource utilization requirement

If time allows, time will be spent identifying system bottlenecks and working towards their elimination.

10 System models

Entity-Relationship Diagram



[online diagramming & design] creately.com

11 System evolution

The system has the potential to evolve into a mature expert knowledge system. Successful evolution is dependent on the following criteria;

- Sufficient expert knowledge to populate the AIML knowledge base
- Improving the answers given by the system through better questioning of the user
- Mature user modeling so that the system has a solid base on which to build new answers

12 Appendices

Use case 4.3.1

Use case

Answer Question

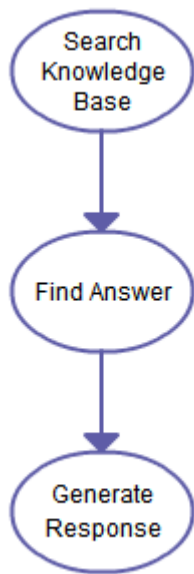
Scope

The scope of this use case is to establish if this is a new or existing user.

Description

This use case describes the simplest interaction with the system.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The system has found/created a profile for the user.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system presents the topic to the bot
2. The answer is known

3. The answer is returned to the calling object for wrapping as a message

Alternate flow

A1 : The answer is unknown

1. The use case 4.3.2 is activated
2. The use case continues at position 3 of the main flow

Exceptional flow

None

Termination

The system has successfully responded to the users question.

Post condition

The system goes into a wait state.

Use case 4.3.2**Use case**

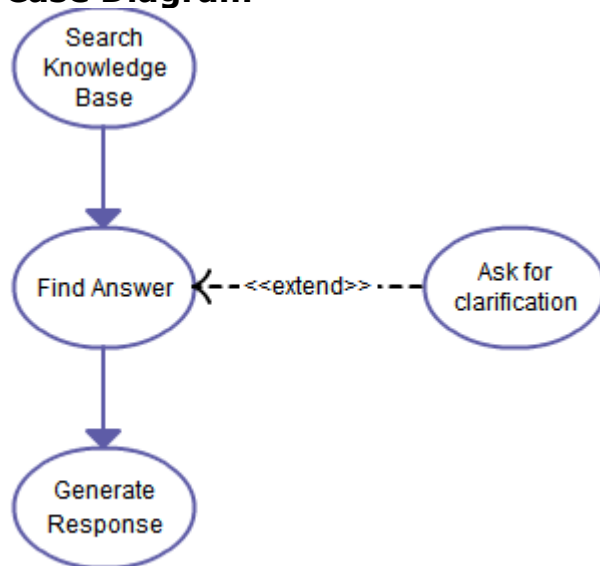
Ask for Clarification

Scope

The scope of this use case is to clarify an ambiguous question asked by the user.

Description

This use case describes what the system will do if the answer is not readily found.

Use Case Diagram**Flow Description**

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The system has found/created a profile for the user.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system presents the topic to the bot
2. The answer is unknown
3. The system asks the user for clarification
4. The system presents the clarification to the bot
5. The answer is returned to the calling object for wrapping as a message

Alternate flow

A1 : The answer is known

1. The use case 4.3.1 is activated
2. The use case continues at position 5 of the main flow

Exceptional flow

None

Termination

The system has successfully responded to the users question.

Post condition

The system goes into a wait state.

Use case 4.3.3**Use case**

Prompt User for further information

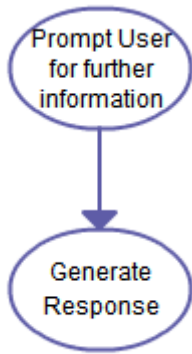
Scope

The scope of this use case is to trigger action by the user if the query has not been completed.

Description

This use case describes what the system will do if the user fails to respond to an earlier query.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The user has already contacted the system.

The system has found/created a profile for the user.

Activation

This use case starts when the system has been waiting for a response from a User (<Actor>) for a pre-determined length of time.

Main flow

1. The system prompts the user for further information

Alternate flow

None

Exceptional flow

None

Termination

The system has prompted the user for further information.

Post condition

The system goes into a wait state

Use case 4.3.4

Use case

Recognition of Knowledge Limitation

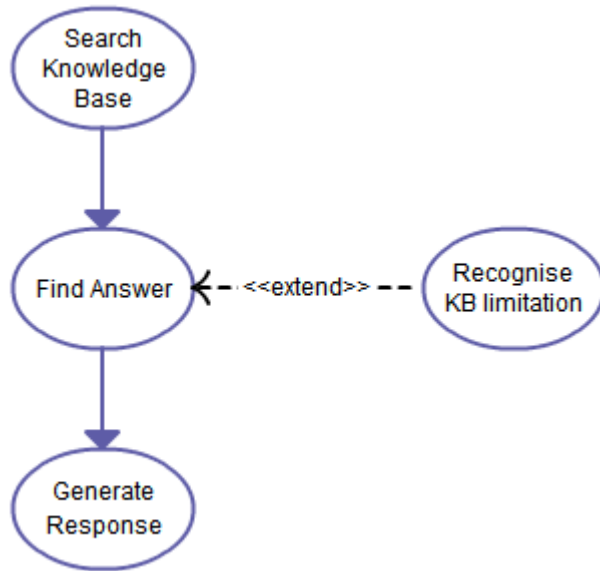
Scope

The scope of this use case is to facilitate the recognition of the limitations of the knowledge base.

Description

This use case describes what the system will do if the answer is not readily found and clarification proves ineffective.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The system has found/created a profile for the user.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system presents the topic to the bot
2. The answer is unknown
3. The system asks the user for clarification
4. The system presents the clarification to the bot
5. The answer is still unknown
6. The system responds to the user notifying him that the answer is unknown
7. The use case 4.3.5 is activated
8. The use case 4.3.6 is activated

Alternate flow

The alternate flow is that the answer is found and is described in Use Case 1.

Exceptional flow

None

Termination

The system notifies the user that the answer is unknown.

Post condition

The system goes into a wait state

Use case 4.3.5**Use case**

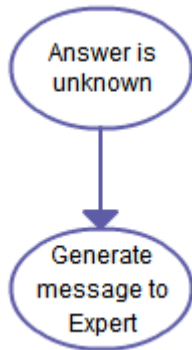
Spawn email to human expert

Scope

The scope of this use case is to notify the human expert.

Description

This use case describes what the system will do if a user question failed to be answered and needs to be addressed.

Use Case Diagram**Flow Description****Precondition**

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The system has found/created a profile for the user.

The answer is not found.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system can't find the answer
2. The system notifies the human expert

Alternate flow

None

Exceptional flow

None

Termination

The system has notified the human expert.

Post condition

The system goes into a wait state

Use case 4.3.6**Use case**

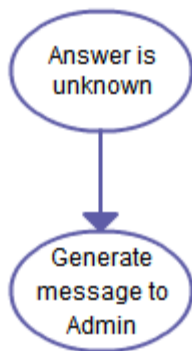
Notify Admin of missing Knowledge

Scope

The scope of this use case is to notify the admin that there is a gap in the Knowledge Base.

Description

This use case describes what the system will do if a user question failed to be answered and needs to be addressed.

Use Case Diagram**Flow Description****Precondition**

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The system has found/created a profile for the user.

The answer is not found.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system can't find the answer
2. The system notifies the admin user that there is a gap

Alternate flow

None

Exceptional flow

None

Termination

The admin user has been notified that there is a gap in the KB.

Post condition

The system goes into a wait state

Use case 4.3.7.6.1**Use case**

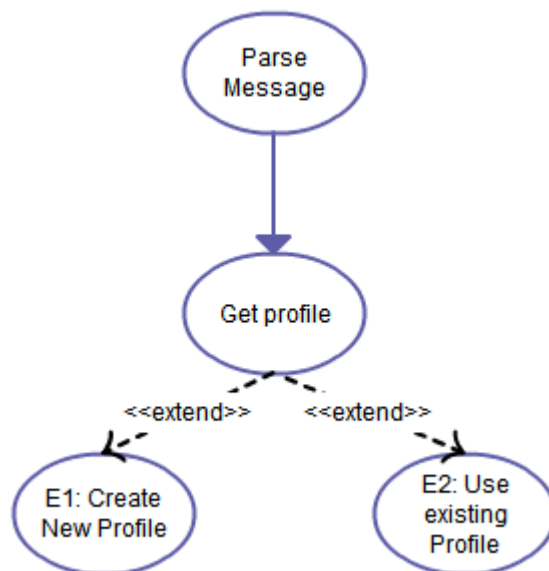
New User

Scope

The scope of this use case is to establish if this is a new or existing user.

Description

This use case describes the initial user interactions with the system.

Use Case Diagram**Flow Description****Precondition**

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

2. The system identifies the user and checks if a profile/model already exists
3. The user (<Actor>) is unknown
4. The system will establish a new profile for this user (See E1)
5. Subsequent activity will be tracked against the profile

Alternate flow

A1 : The user is known

3. The system identifies the user and checks if a profile/model already exists
4. The user (<Actor>) is known
5. The use case continues at position 4 of the main flow

Exceptional flow

None

Termination

The system knows the user.

Post condition

The system goes into a wait state

Use case 4.3.7.6.2

Use case

Existing User

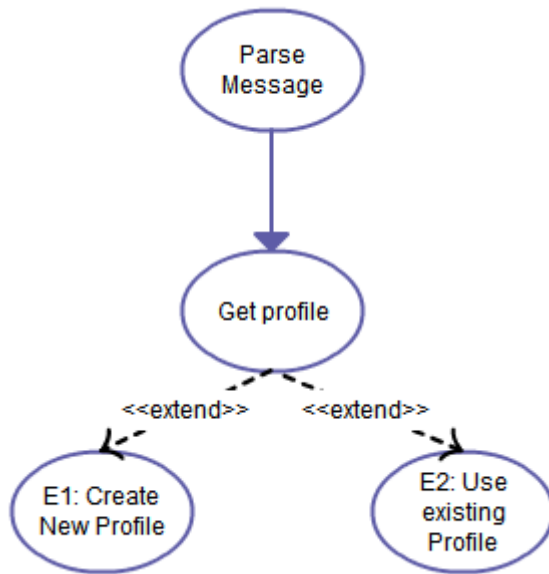
Scope

The scope of this use case is to establish if this is a new or existing user.

Description

This use case describes the initial user interactions with the system.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

Activation

This use case starts when a User (<Actor>) sends an email to the system.

Main flow

1. The system identifies the user and checks if a profile/model already exists
2. The user (<Actor>) is known
3. Subsequent activity will be tracked against the known profile

Alternate flow

A1 : The user is known

1. The system identifies the user and checks if a profile/model already exists
2. The user (<Actor>) is unknown
3. The system will establish a new profile for this user (See E1)
4. The use case continues at position 3 of the main flow

Exceptional flow

None

Termination

The system knows the user.

Post condition

The system goes into a wait state

Use case 4.3.7.6.3

Use case

Establish knowledge about user

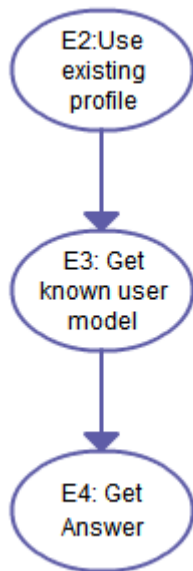
Scope

The scope of this use case is to show that information established in earlier interactions is available to the system.

Description

This use case describes the initial user interactions with the system.

Use Case Diagram



Flow Description

Precondition

The system is in a ready state.

The Gmail email service is available.

The user has access to the internet.

The message has been parsed as only the user and topics are useful at this point.

The user profile has been found.

Activation

This use case starts when a profile has been established by the system.

Main flow

1. The system gets the user model (E3)

2. The question is answered in the context of the known user model (E4)

Alternate flow

A1 : The user model is empty

1. The system adds the question to the model
2. The use case continues at position 2 of the main flow

Exceptional flow

None

Termination

The system knows the user.

Post condition

The system goes into a wait state